

Note: in which the data should be stored.

137

Eg:  $\text{int } a = 10$   
 $\text{int } *b;$

$\text{float } c = 10.5;$

$\text{float } *d;$

$b = &a;$

$d = &c;$

① address will take 4 bytes. (in program we assume address 4 byte).  $\rightarrow$  main block

$\text{pf}(a) = ?$  10

$\rightarrow \text{size of } (a) = 2B$  || size of a int 2 B

$\rightarrow \text{size of } (c) = 4B$  || size of a float = 4 B

$\rightarrow \text{size of } (b) = 4B$

$\rightarrow \text{size of } (d) = 4B$

$\rightarrow \text{size of } (*a) = 4B$

$\rightarrow \text{size of } (*c) = 4B$

$\rightarrow \text{size of } (*b) = 4B$

$\rightarrow \text{size of } (*d) = 4B$

■ **Pointer to array** :- Array is collection of homogeneous elements.

→ Mem for array elements is continuously allocated.

will be allocated

① **Array name** / w/o subscript contains base address of the array allocated by the compiler.

array allocated by the compiler is a constant and it can't be changed.

② This base address is a constant and it can't be changed.

③ **void main()**

$\text{int } a[5] = \{10, 20, 30, 40, 50\};$

$\text{int } *p1; \quad \text{int } *p2 = &a[0];$

$p1 = &a[0];$   
 $\text{int } *p1;$   
 $p1 = a;$

$\text{++ } p1;$

$\text{++ } *p1;$

$\text{pf}(" \%d", *p1);$

$\text{pf}("%d", p1);$

associativity of unary operator  
right to left.

Ans: 30.29

$\text{pf}(a) = 1000$

$\text{pf}(p1) = 1000$

Notes: ① When the pointer variable is  $\text{int}$  then it will w.r.t. size of the elements to which it is pointing to.

② Unary Operator associativity is R  $\rightarrow$  L.

(3) See the above explanation pls, as both are pointers to array but  $p_1$  is a variable &  $a$  is a pointer, constant

1381

$\rightarrow p_1$   
 $\rightarrow - p_1$   
 $\rightarrow + a$   ~~$\rightarrow X$~~   
 $\rightarrow - a$   ~~$\rightarrow X$~~

(4) Array name is given as constant pointer to let  $c_2$  i.e. the only source in the form how the array's contents can be accessed using  $*$  operator is  $a[2]$  which can be written as  $*a + 2$

Void main()

int  $a[5] = \{10, 20, 30, 40, 50\}$   
 $\text{printf}(*, .d \%d \%d \%d \%d)$ ,  $a[2], 2[a], * (a+2), *(2+a)$

$a[2]$	$a[2]$
$* (a+2)$	$* (a+2)$
$* (1000 + 2)$	$* (2+a)$
$* (1004)$	$2[a]$
30	0 10 20 30 40 50

Point for Arithmetic :-

Void main()

int  $a[5] = \{10, 20, 30, 40, 50\}$

before  $\star p_1 = a[i]$

int  $\star p_2 = a + 4;$

$i = \{0, 1, 2, 3, 4\}$

$i = \{1000, 1002, 1004, 1006, 1008\}$

$i = \{1000 + 4 * 2, 1002 + 4 * 2, 1004 + 4 * 2, 1006 + 4 * 2, 1008 + 4 * 2\}$

$i = \{1008, 1012, 1016, 1020, 1024\}$

$i = \{1008, 1012, 1016, 1020, 1024\}$

① We can add integer constant to a pointer, it means skipping the elements in forward direction.

Eq:  $p_1 + 2$

$\underbrace{1000}_{p_1} + 2 = 1004$

$p_1 = \{1000\}$

P.S. All data

② We can subtract integer constant from pointer it means skipping the elements in backward direction.

$1008 - 2 = 1006$

$$1008 - 2 * 2$$

$$= 1008 - 4$$

$$= 1004$$

139

- ③ We can subtract two pointers, this is allowed only when both are pointing to same array &  $p_2 > p_1$ .  
This operation will give  $N$  # elements b/w from  $p_2$  to  $p_1$ .

$$\text{e.g. } P_2 - P_1 \leftarrow \text{ to find}$$
$$1008 - 1000 = 8 = \frac{8}{2} = 4$$

$$\boxed{P_2 > P_1}$$
$$1008 - 1008 = 0 = \frac{0}{2} = 0$$

- ④  $P_1 + 2 \cdot 5$  ✗ We can't add floating point number to two pointers b/c it was no meaning.
- ⑤ We can't perform  $+, *, /$  b/w two pointers.  
it was no meaning.

$$P_1 + P_2 \otimes$$

$$\text{so } P_1 * P_2 \otimes$$

$$P_1 / P_2 \otimes$$

Eg: void main() {  
int arr[5] = {10, 20, 30, 40, 50};  
int \*p1 = arr;  
int \*p2 = arr + 4;

$$\text{int } *p_1 = a;$$

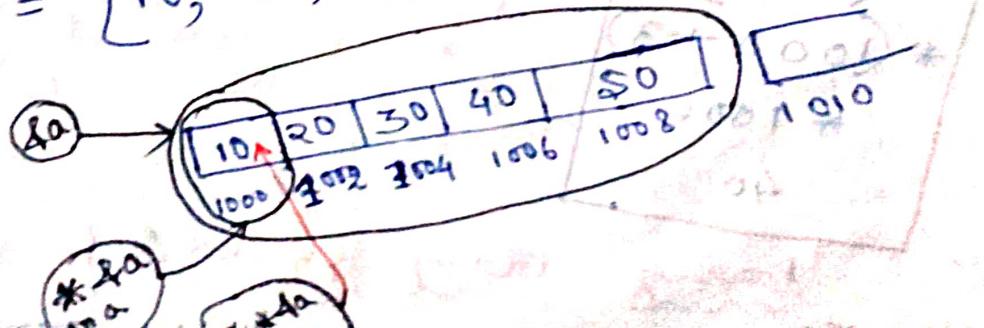
$$\text{int } *p_2 = a+4;$$

int pointf((u, d), p2 - p1);

$$\text{out: } 1008 - 1000 = 8 = \frac{8}{2} = 4.$$

One dimensional Array

$$\text{int arr[5]} = \{10, 20, 30, 40, 50\}$$



Expression	Point Value	Type	Meaning
1. $\&a$	1000	int*	Base address of entire one dimensional array.
2. $*\&a$	1000	int*	Base address of 0th element of the array.
3. $a$	1000	int*	Base address of the 0th element of the array.
4. $*a$	10	int	Value of the 0th element of the array.
5. $\&a+1$	1010	int*	Base address of the next contiguous 1D array.
6. $a+1$	1002	int*	Base address of the 1st element of the array.
7. $*a+1$	11	int	Value of 0th element.
8. $**a$	error U2. Value	?	outside the array now
9. $a+3$	1006	int*	base address of the 3rd element of the array.
10. $*(\&a+3)$	40	int	Value of the 3rd element of array.
11. $a[3]$	40	int	Value of the 3rd element of the array.

$$\begin{array}{|c|} \hline *(\&a+3) \\ *(\&1000+3) \\ *\&1006 \\ 40 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 1000 | 0x1000 | 1001 | 0x1001 \\ \hline \end{array}$$

Ques-2 Passing array pointer to function:

Note: If u want to pass all the array elements to a function, then it is not a good idea to pass the array

elements, instead by simply passing locus address of the array we can access all the array elements.

Eg: void main()

int a[5] = {10, 20, 30, 40, 50};

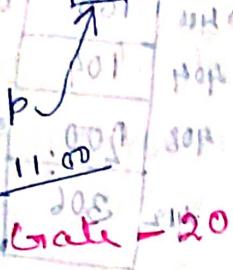
display(a, 5);

number of arguments and memory location of the variable are passed to function.

and just void display(int \*p, int n) {  
int i; for(i=0; i<n; i++) {  
printf("%d", p[i]);

} what is the o/p printed?

10 20 30 40 50



Q) Gate - 20

①

void main() {  
int a[5] = {12, 7, 13, 4, 6};  
int b = f(a, 5);  
printf("%d", b);

What is the

o/p printed?

int f(int \*a, int n) {

if (\*a == 0) {  
return 0;  
else if (\*a \* 1.2 == 0) {  
return 1;  
else if (\*a \* 1.2 != 0) {  
return 2;

15  
else if (\*a \* 1.2 == 0) {  
return 3;  
else if (\*a \* 1.2 != 0) {  
return 4;  
else if (\*a \* 1.2 == 0) {  
return 5;  
else if (\*a \* 1.2 != 0) {  
return 6;  
else if (\*a \* 1.2 == 0) {  
return 7;  
else if (\*a \* 1.2 != 0) {  
return 8;  
else if (\*a \* 1.2 == 0) {  
return 9;  
else if (\*a \* 1.2 != 0) {  
return 10;  
else if (\*a \* 1.2 == 0) {  
return 11;  
else if (\*a \* 1.2 != 0) {  
return 12;  
else if (\*a \* 1.2 == 0) {  
return 13;  
else if (\*a \* 1.2 != 0) {  
return 14;  
else if (\*a \* 1.2 == 0) {  
return 15;

else {  
a = f(a + 1, n - 1);

return a;

12 7 13 4 11 16  
1000 1002 1004 1006 1008 1010

$$1 - f(1000, 1) = 15$$

$$15 - f(1004, 4) = 3$$

$$3 - f(1008, 8) = 6$$

$$6 - f(1012, 0) = 4 = 13 - f(1005, 5)$$

$$4 = 14 - f(1008, 9)$$

## Array of pointers:

Just like we had a normal integer array, similarly we have array of pointers.

Eg: int  $a[5]$ , \* $a[5]$ . 

- Between Operator & Operand Operand has high precedence.
  - Between \* and Braces [ ] has high precedence.

Eg: main() { } Statement outside in main

① int a=10, b=20, c=30, d=40;

int \*p[4];

int \*\*\* p[2] = P;

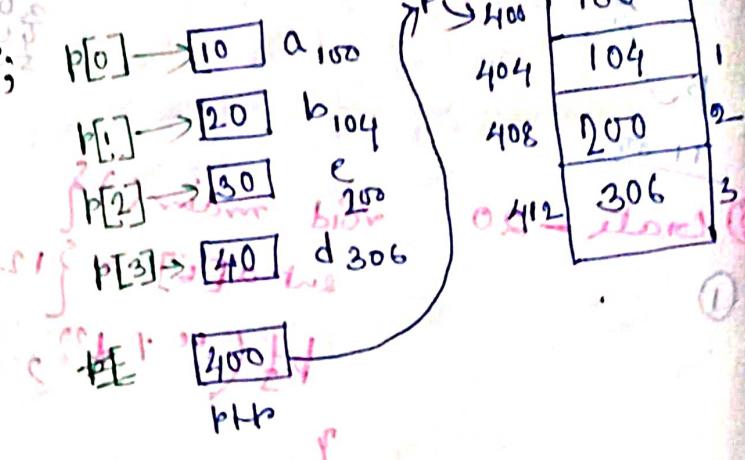
$$\phi[0] = \vartheta a;$$

$$b[1] = 4b;$$

$$k_{\Gamma 07} = 8.2$$

$$P[2] = 4.5,$$

$$b[3] = 4d;$$



Eg ①

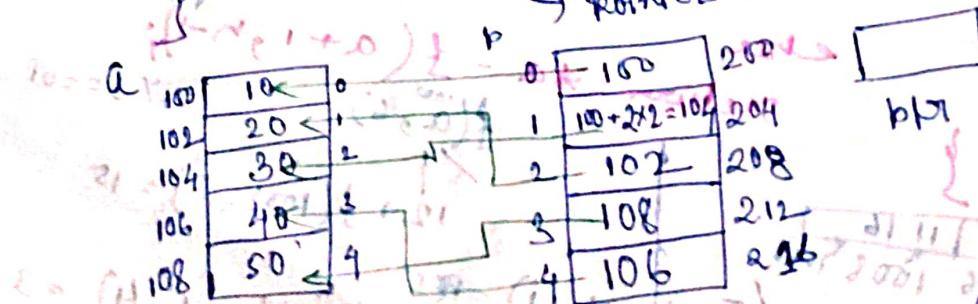
$E_d$  ② remain { }  $\left\{ \text{for this set this} \right\}$  - we

int a[5] = {10, 20, 30, 40, 50};

int \*b[5] = {a, a+2, a+1, a+4, a+3};

$$(int) f * * p(x) = f$$

~~our DE~~ ~~\* \* p[0] = P,~~ ~~↓ pointer~~ which point arr ~~pointer.~~



pointer stores  
address which is  
of 4 bytes.

b To variable → .reside value

③ void main() {

④ int a[5] = {0, 102, 3, 4};

int \*b[5] = {a, a+1, a+2, a+3, a+4};

int \*\*c[5] = b;

↳ ~~Point to the address of the first element of the array a~~

$\text{pf}(\text{b}[0] - \text{p}, * \text{b}[0] - \text{a}, ** \text{p}\text{b})$ ;

$\text{pf}(* \text{b}[0] + 1; \text{b}[0] - \text{a}, * \text{b}[0] - \text{a})$ ;

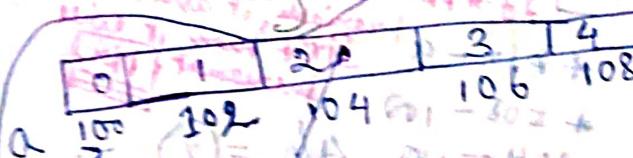
$\text{pf}(* \text{b}[0] - \text{p}, * \text{b}[0] - \text{a}, * \text{b}[0] - \text{a}, ** \text{p}\text{b})$ ;

$* \text{b}[0] + 1;$

$\text{pf}(* \text{b}[0] - \text{p}, * \text{b}[0] - \text{a}, * \text{b}[0] - \text{a}, * \text{b}[0] - \text{a}, ** \text{p}\text{b})$ ;

$* \text{b}[0];$

$\text{pf}(* \text{b}[0] - \text{p}, * \text{b}[0] - \text{a}, * \text{b}[0] - \text{a}, * \text{b}[0] - \text{a}, * \text{b}[0] - \text{a}, ** \text{p}\text{b})$ ;



O/P: 1.  $\text{p}\text{b} - \text{p} = 504 - 500$

[from  $\text{p}\text{b} = 4$  before p]

(2)  $\frac{4}{4}$  (1)

2.  $* \text{p}\text{b} - \text{a}$

$\text{p}\text{b}$  variable gives an error  
 $* 504 \rightarrow 102 - 100 = 2$  both are pointing to different elements to different arrays

3.  $** \text{p}\text{b} = \text{p}\text{b} \text{b}$ ,

$\frac{500}{102}$

500	100 100
504	102 102
508	104
512	106
516	108

500

504

508

512

516

518

520

524

528

532

536

540

544

548

552

556

560

564

568

572

576

580

584

588

592

596

600

604

608

612

616

620

624

628

632

636

640

644

648

652

656

660

664

668

672

676

680

684

688

692

696

700

704

708

712

716

720

724

728

732

736

740

744

748

752

756

760

764

768

772

776

780

784

788

792

796

800

804

808

812

816

820

824

828

832

836

840

844

848

852

856

860

864

868

872

876

880

884

888

892

896

900

904

908

912

916

920

924

928

932

936

940

944

948

952

956

960

964

968

972

976

980

984

988

992

996

1000

1004

1008

1012

1016

1020

1024

1028

1032

1036

1040

1044

1048

1052

1056

1060

1064

1068

1072

1076

1080

1084

1088

1092

1096

1100

1104

1108

1112

1116

1120

1124

1128

1132

1136

1140

1144

1148

1152

1156

1160

1164

1168

1172

1176

1180

1184

1188

1192

1196

1200

1204

1208

1212

1216

1220

1224

1228

1232

1236

1240

1244

1248

1252

1256

1260

1264

1268

1272

1276

1280

1284

1288

1292

1296

1300

1304

1308

1312

1316

1320

1324

1328

1332

1336

1340

1344

1348

1352

1356

1360

1364

1368

1372

1376

1380

1384

1388

1392

1396

1400

1404

1408

1412

1416

1420

1424

1428

1432

1436

1440

1444

1448

1452

1456

1460

1464

</div

0	1	2	3	4
100	102	104	106	108

	100	100 ← X	569	568
0	100	104 ← X	569	519
1	102	508 ←	560	
2	104	548 ←	561	
3	106	588 ←	571	
4	108	516		

① 1111++  
500

$$\textcircled{2} \quad \frac{504 - 500}{4} = \textcircled{1}$$

③  $*\text{ptr} - \text{a}$  [Binary Operator was illegal]  
 $\text{ptr} = \text{arr}$   
 $102 - 102$   
 $\frac{2}{2} = 1$  [Both are pointing to same array,  
so subtraction is allowed]  
[From  $*\text{ptr}$  to  $\text{arr}$ ) how many elements  
the array contains →]

(4)  $\text{***PbO}_2$   
 $\text{---} \overset{\text{504}}{\underset{\text{102}}{\text{---}}}$

$$\textcircled{S} \quad \begin{array}{c} \text{* } bkg + t \\ \hline \text{* } BBS \\ \text{---} \\ 102 \end{array}$$

$$\textcircled{6} \quad p1 = p2 - \frac{508 - 500}{8/4} = \textcircled{2}$$

⑧ 米粉粥; ②

$$\begin{array}{r} 568 \\ \times 104 \\ \hline \end{array}$$

11. 3 12. 3

108  
⑨ 1913 2

⑨ main () { }

int a[5] = {

ent \* p[5] =

卷之四十一

$$P_f(P_{\text{in}} - P),$$

\* + + \* PFT

$$P \subseteq P(\alpha - b, *$$

	10	20	30	40	50
100	100	102	104	986	108

104	500
104, 102	504
106	508
100	512
108	516

504  
500

[Univ. operator freedom right to left]  
But this is post increment, after executing the line  
the ans will be ~~intercepted~~ 1,8020

if  $x = \frac{1}{2} \log \left( \frac{1 + \sqrt{5}}{2} \right)$  then  $\theta = 56.8^\circ$

~~50 125 H 36 50/65  
102 104 118~~

0	108	508	584
1	106	506	585
2	104	508	582
3	104	506	582
4	104	506	582

First element

- If it is 1D array, then 1st element means base address of 0th element.  
Eg: int a[5];  
 $a \Rightarrow \text{int } *$
- If it is a 2D array then 1st element is base address of 0th 1D array.
- If it is 3D array, then first element means base address of 0th 2D array.

Eg: int a[3][4][5];  
 $a \Rightarrow \text{int } *[*][*]$

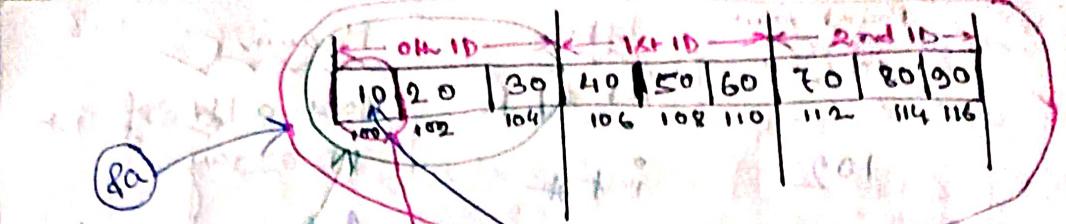
2D array: → Eg: int a[3][3] = {10, 20, ..., 90}

- NOTE:
- ① A good example for the two dimensional array is pointer.
  - ② The above example represents 3 rows & 3 columns.
  - ③ In the 2D array,  $a[i]$  represent constant base address of  $i$ th row allocated by the compiler & it can't be changed.
  - ④ If we perform  $a[i] = 2000$ ; This gives an error, coz we can't change the ~~constant~~ base address of  $i$ th row.
  - ⑤ The 2D array stored in the memory in the form of multiple 1D arrays.
  - ⑥ Array name without subscript gives constant base address of 0th or 1D array for pointer.
  - ⑦ If we perform  $a[i][j] = 20$ ; //  $i$ th row,  $j$ th column will replace the value of 20.

Other		1st ID		2nd ID			
10	20	30	40	50	60	70	80
101	102	104	106	108	110	112	114

118

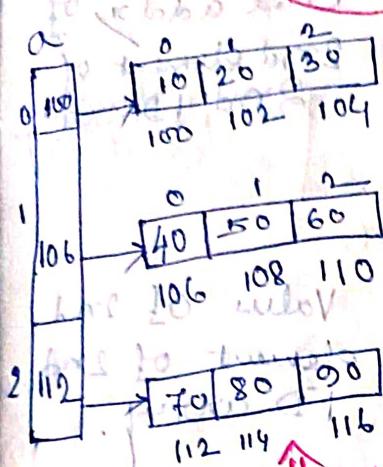
Ques: answer for  
Tenth question



18

147.

Physical view.



Logical view

Expression      Point value      Type      Meaning

1)      &amp;a      100      int \* [ ]

DF: 15.05.21  
 $(c + (c+0) *) *$  $(c + (c+001) *) *$ 

Expression	Point value	Type	Meaning
1) &a	100	int * [ ]	Base add <sup>r</sup> y of entire 2D array.
2) *a	100	int * [ ]	Base add <sup>r</sup> y of 0th 1D array.
3) a	100	int * [ ]	Base add <sup>r</sup> y of 0th 1D array.
4) *a	100	int *	Base add <sup>r</sup> y of 0th element of 0th 1D array.
5) **a	100	int	Value of 0th element of 0th 1D array.
6) a[0]	100	int *	Base add <sup>r</sup> y of 0th element of 0th 1D array.
7) *(a+0)	100	int *	Base add <sup>r</sup> y of 0th element of 0th 1D array.
8) &a+1	118	int * [ ]	Next continuous 2D array.

Expression	Point value	Type	Meaning
(8) $a + 1$	106	int *	Base add <sup>y</sup> of 1st 1D array
(9) $*a + 1$	102	int *	Base add <sup>y</sup> of 1st element of 0th 1D array
(10) $*(*a + 2) + 2$	116	int *	Base add <sup>y</sup> of 2nd element of 2nd 1D array.
$\equiv *(*a + 2 * 6) + 2$			
$= *112 + 2$			
(11) $*(*a + 2) +$ $*(*(*a + 2) + 2)$	90	int	Value of 2nd element of 2nd 1D array.

$a[2][2]$   
 $*(*(*a + 2) + 2)$   
 $*(*(*100 + 2) + 2)$   
 $\downarrow$   
 $*(*(*112) + 2)$   
 $*(*112 + 2)$   
 $*116$   
 90

[one bracket means one pointer]

Eg:  $a[1][1]$

$*(*(*a + 1) + 1)$  → pointing to the entire 0th 1D array  
 $*(*(*100 + 8) + 1)$   
 $*(*106 + 1)$   
 $*106$   
 $*108$   
 50

$*(*(*112) + 2)$  → pointing to the entire 1st 1D array  
 $*(*112 + 2)$   
 $*114$   
 $*115$   
 51

$*(*(*116) + 2)$  → Value of the 1st element of 1st 1D array.  
 $*117$   
 52

Explain

110

int \* -

B. A. 83

2nd element of 1st 1D array

13)  $*(a+1)+2$

error (most)  
local variable  
 $a \leftarrow \{ \text{some} \}$

?

different location or pointing outside array

GATE

void main()

int a[5][3] = {10, 20, 30, ..., 150};  
if ("d", ((a == a[0]) & (a[0] == \*a))),

What is the output printed?

(a) 1

(b) 0

(c) compilation error

(d) Run time error

During comparison, the  
point value will be compared  
The type isn't mandatory.  
Here, the type is same.  
when + op is done.  
Eg.  $(a+)$ .

$$a[0] = *(a+0)$$

\*a

(a) 1

int \*A[10]; → pointer array

int B[10][10]; → 2D array

(b)

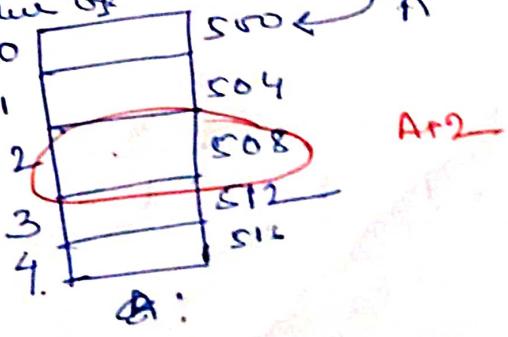
of the following expressions —

(i) A[2] (ii) A[2][3] (iii) B[1] (iv) B[2][3]

Which of the following expression will not give compile time error, if we use them as left side of assignment statement in C program?

(i) ~~A[10]~~ int \* A[10] → A is a pointer array which contains 10 int.

(ii) ~~A[2]~~ A[2] = \* [A + 2] → pointing to value of 2nd element. 0



(ii) A[2][3] = \* [\*(A + 2) + 3]  
\* 508  
5. N. + 3 → error

(iii) B[1] = \*(B + 1) → pointing to 0th element of 1st 1D array.

(iv) B[2][3] = \* (\* (B + 2) + 3) → pointing to 3rd element of 2nd 1D array.

(i) & (iii) will give compile time error. (ii) & (iv) need runtime error.

declaration - 2:

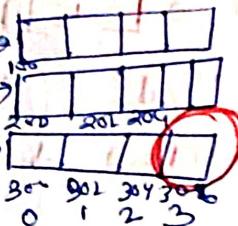
int \* A[10];

- left to right order, operand must have higher precedence than operator.
- Operand having higher precedence than Operator.
- $A + \frac{1}{2}$  first operator  $\Rightarrow$  [ ] has higher precedence.
- A is an array of pointers of type integer.

(ii)

$A[2][3]$

$A \rightarrow 800$



compile time  
error

If we will not give any compile time error.

H.W.

①  $\text{int } a[4][5] = \{1, 2, 3, 20\};$

\* Implement all expressions,  $B[A] = 100$ , size of elements = 2B.

Pointers stored  $\rightarrow [0]A + \text{bias}$

Size of process  $\rightarrow 4[1]T[1] + \text{bias}$

Let's take  $[0]A$  as base address with bias  $\rightarrow$   $[0]T[1] + [0]A$ .  $[0]T[1] + [0]A$   $\rightarrow$   $[0]T[1] + [0]A + \text{bias}$

pointer stored  $\rightarrow [0]A + \text{bias}$

$A \rightarrow 802$   $\rightarrow$   $[0]A + \text{bias}$

pointer stored  $\rightarrow [0]A + \text{bias}$

$$[0] + [A] * = [0]A$$

$$[0] + [I + A] * = [0] + [I]A$$

$802 *$

Process,  $\text{size} + \text{bias}$

Process  $\rightarrow 10$   $\rightarrow$   $(1 + 9) * = [0]A$

Process  $\rightarrow 10$   $\rightarrow$   $(1 + 9) * = [0]A$

2

$$\text{int } a[4][5] = \{1, 2, 3, i \in [20]\}; \text{ B.A} = 100, \text{ size of elements} = 2B.$$

$$\textcircled{a} \quad * (a+2) +$$

三

203

201 B.

$A = 100$ , size of elements =  $2B$ .

1

$$\textcircled{d} \quad \#(a+1) + 3. \quad \textcircled{e}) \quad a+3 - \frac{5}{1}(*a+2)$$

$$\textcircled{a} \quad * (a+2)^{+1}$$

pointing to the 1st element of 2nd 1d array,  $\{10 \ 20 \ 30\}$

to towels w10 for 10 min  
w10 for 10 min

~~Widespread species  
over forested areas of  
the tropics and subtropics~~

~~12.1~~ ~~to 1660~~ ~~2008~~  
~~12.1~~ ~~to 166A~~ ~~2008~~  
new 18.110 ~~to~~ ~~1660~~

To work out

MA 11 340 to turn 3D Answer

① force of w.t. ~

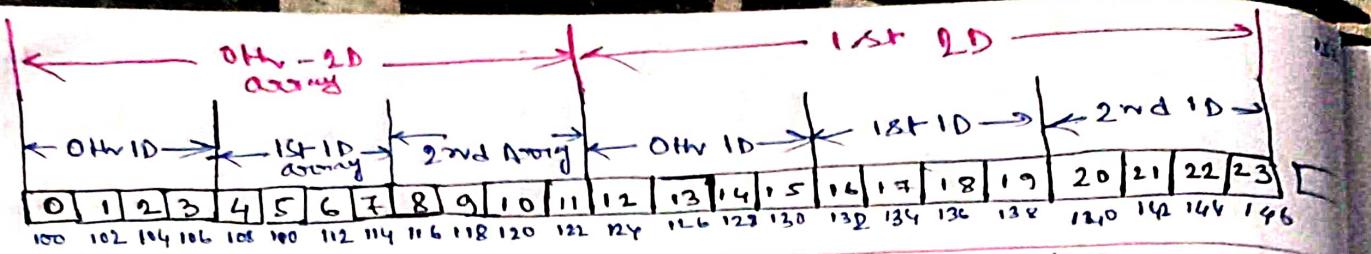
⑤ ~~Average~~ mean ~~are~~  
~~for~~ ~~the~~ ~~first~~ ~~2~~ ~~is~~ ~~20~~ ~~44~~

② 3D wavy brain

the form of the

Eq. 1:  $\int \frac{1}{x^2} dx$

W 13  
19000 13  
19000 13



Expression	P.V.	Type	Meaning
1) $\&a$	100	$\text{int}^*[\cdot][\cdot][\cdot]$	Base Address of entire 3D array.
2) $*\&a$	100	$\text{int}^*[\cdot][\cdot]$	Base address of 0th 2D array.
3) $\&a$	100	$\text{int}^*[\cdot][\cdot]$	
4) $*a$	100	$\text{int}^*[\cdot]$	Base address of 0th 1D array of 0th 2D array.
5) $**a$	100	$\text{int}^*$	B.A. of 0th element of 0th 1D array of 0th 2D array.
6) $***a$	0	int	Value of 0th element of 0th 1D array of 0th 2D array.
7) $***a$	error	?	suffixing unknown location or outside coverage error.
8) $\&a+1$	147	$\text{int}^*[\cdot][\cdot][\cdot]$	Base address of next 3D array.
9) $a+1$	124	$\text{int}^*[\cdot][\cdot]$	Base address of next 1st 2D array.
10) $*a+1$	108	$\text{int}^*[\cdot]$	Base address of 1st 1D array of 0th 2D array.
11) $**a+1$	102	$\text{int}^*$	Base Address of 1st element of 0th 1D Array of 0th 2D array.
12) $*(\&a+1)+2$	140	$\text{int}^*[\cdot]$	B.A. of 0th element of 2nd 1D array of 1st 2D array.
13) $*(*(\&a+1)+1)+2$	140	$\text{int}^*$	B.A. of 0th element of 2nd 1D array of 1st 2D array.
14) $*(*(\&(a+1)+1)+1)$	18	int	Value of 2nd element of 1st 1D array of 1st 2D array.